Symbolic Encoding of String Lengths

Fang Yu Tevfik Bultan Oscar H. Ibarra

Dept. of Computer Science
University of California Santa Barbara, USA
{yuf, bultan, ibarra}@cs.ucsb.edu

October 3, 2008

1 Motivation
- String Analysis + Integer Analysis
- What is Missing?

2 Examples
- What is Its Length?

3 Implementation
- Symbolic Encoding

4 Conclusion

Motivation

We aim to develop a verification tool for analyzing systems with **strings and integers**.

We propose a **composite analysis** that combines *static string analysis* and *arithmetic analysis*.

Arithmetic Analysis

Arithmetic Analysis: At each program point, symbolically compute all possible states of **integer** variables.

These infinite states are symbolically over-approximated as a Presburger arithmetic (linear arithmetic) formula and represented as an **arithmetic automaton** [Bartzis and Bultan, CAV03].

Integer analysis is widely used to detect **buffer overflows** by representing lengths of string variables as integer variables.

String Analysis

Static String Analysis: At each program point, statically compute all possible values that **string** variables can take.

These values are specified as the language that is accepted by a **string automaton** [Fang et al. SPIN08].

Static string analysis is widely used to detect **web vulnerabilities** like SQL Command Injection and Cross Site Scripting (XSS) attacks.

What is Missing?

A motivating example from trans.php, distributed with MyEasyMarket-4.1.

- 1:<?php
- 2: $www = $_GET["www"];
- 3: $l_otherinfo = "URL";
- 4: $www = ereg_replace("[^A-Za-z0-9 ./-@://]","",$www);
- 5: if(strlen($www) < $limit)
- 6: echo "<td>" . $l_otherinfo . ": " . $www . "</td>";
- 7:?>

What is Missing?

If we perform **arithmetic analysis** solely, after line 4, we lose the length of $www.

- 1:<?php
- 2: $www = $_GET["www"];
- 3: $l_otherinfo = "URL";
- **4: $www = ereg_replace("[^A-Za-z0-9 ./-@://]","",$www);**
- 5: if(strlen($www) < $limit)
- 6: echo "<td>" . $l_otherinfo . ": " . $www . "</td>";
- 7:?>

What is Missing?

If we perform **string analysis** solely, at line 5, we cannot check the branch condition.

- 1:<?php
- 2: $www = $_GET["www"];
- 3: $l_otherinfo = "URL";
- 4: $www = ereg_replace("[^A-Za-z0-9 ./-@://]","",$www);
- **5: if(strlen($www) < $limit)**
- 6: echo "<td>" . $l_otherinfo . ": " . $www . "</td>";
- 7:?>

What is Missing?

We need to connect
the information in the string automata and the arithmetic automata.

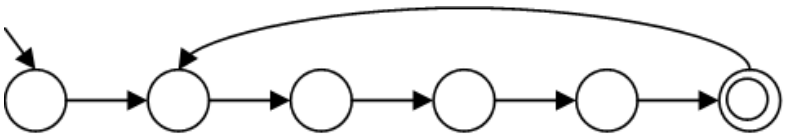
We do this by constructing length automata that accept the **length** of the language of given string automata. The length can be either in unary or binary (from the least significant bit) encoding.

The construction algorithm is detailed in the abstract. Here, I will just show some examples that we automatically compute.

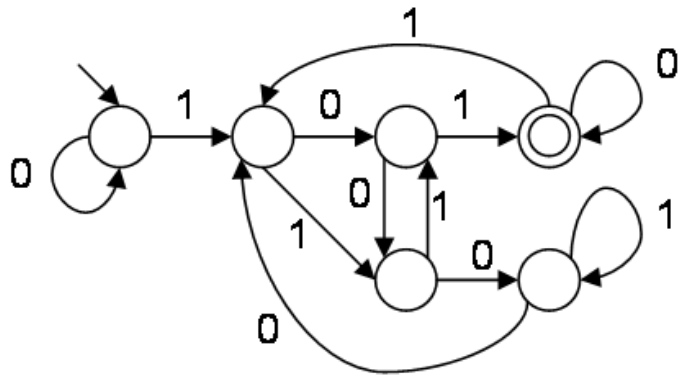
An Example of Length Automata

Consider a string automaton that accepts $(great)^+$.
The length set is $\{5 + 5k | k \geq 0\}$.

- 5: in unary 11111, in binary 101, from lsb **101**.
- 1000: in binary 1111101000, from lsb **0001011111**.



(a) Unary

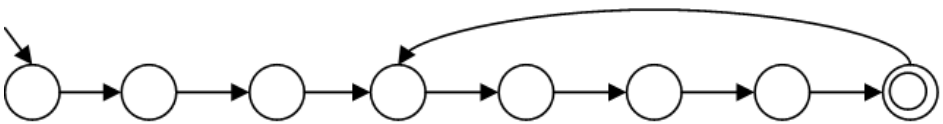


(b) Binary

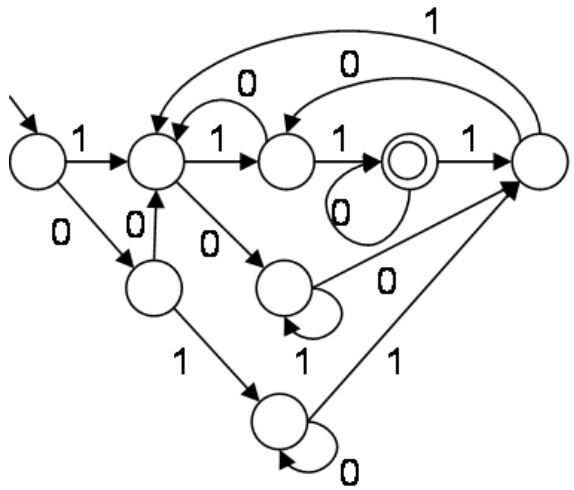
Another Example of Length Automata

Consider a string automaton that accepts $(great)^+cs$.
The length set is $\{7 + 5k | k \geq 0\}$.

- 7: in unary 1111111, in binary 1100, from lsb **0011**.
- 107: in binary 1101011, from lsb **1101011**.
- 1077: in binary 10000110101, from lsb **10101100001**.



(c) Unary



(d) Binary

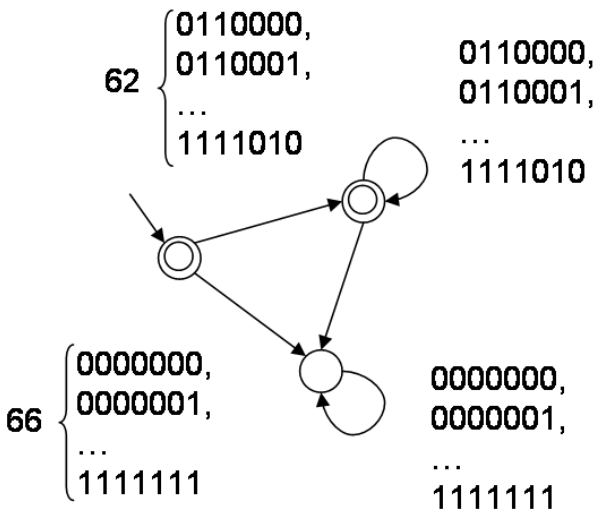
Implementation

We implemented both string and arithmetic automata symbolically by using the MONA DFA Package. [Klarlund and Møller, 2001]

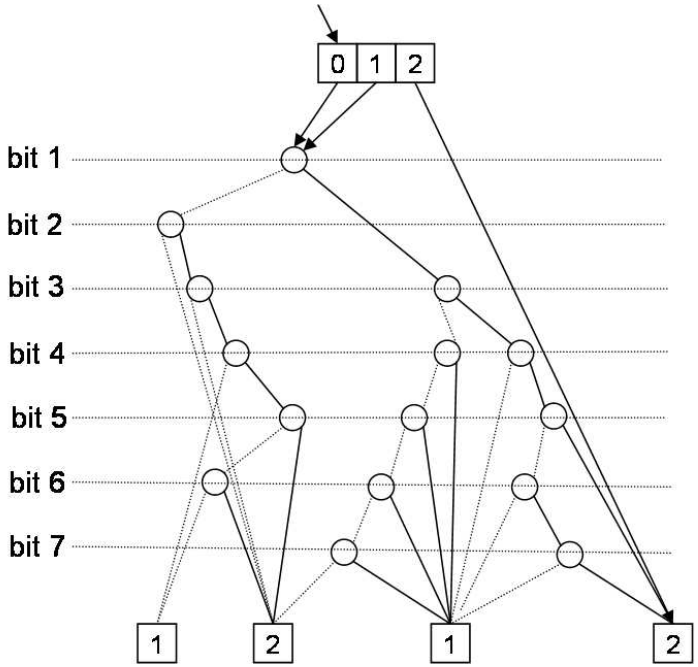
- Compact Representation:
  - Canonical form and
  - Shared BDD nodes
- Efficient MBDD Manipulations:
  - Union, Intersection, and Emptiness Checking
  - Projection and Minimization
- Cannot Handle Nondeterminism:
  - We used dummy bits to encode nondeterminism

Symbolic Encoding

A string automaton that accepts [A-Za-z0-9]* (in ASC II).



(e) Explicit Representation



(f) Symbolic Representation

Final Remarks

How to put length automata in:

- The binary length automaton of a string automaton is a **one-track arithmetic automaton**
- We construct a **multi-track arithmetic automaton** to represent the states of the lengths of string variables.
- This arithmetic automaton is **updated according to the length automata** that we compute from string automata at each iteration.

Final Remarks

Current Works:

- Implement the composite analysis tool that combines the string and arithmetic analyses.
- Use this tool to detect buffer overflows in legacy C systems.

Thank you for your attention.

Questions?

More Information:
http://www.cs.ucsb.edu/∼bultan/vlab
http://www.cs.ucsb.edu/∼yuf